PCT/AU03/01341

Rec'd 13 APR 2005 10/531229

REC'D 27 OCT 2003

WIPO PCT

PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Patent Office
Canberra

I, JULIE BILLINGSLEY, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. 2002952106 for a patent by SILVERBROOK RESEARCH PTY. LTD. as filed on 15 October 2002.

COMMONWEALTH OF AUSTRALIA
AUSTRALIA
PATENT OFFICE

WITNESS my hand this
Twenty-fourth day of October 2003

J. Billingsley

JULIE BILLINGSLEY
TEAM LEADER EXAMINATION
SUPPORT AND SALES

AUSTRALIA

Patents Act 1990

Provisional Specification

for an invention entitled:

METHODS AND SYSTEMS (NPW008)

The invention is described in the following statement: -

METHODS AND SYSTEMS (NPW008)

Improving Recognition Accuracy Using Form Information

Jonathon Leigh Napper and Paul Lapstun
{jon.napper,paul.lapstun}@silverbrookresearch.com

Silverbrook Research Pty Ltd
393 Darling Street, Balmain NSW Australia
2 October 2002

1 Introduction

Recent advances in pattern classification have allowed sophisticated applications to be developed that can recognize natural language input such as speech or handwriting. These applications allow users to communicate with the system in a natural and convenient way, and permit the automation of tasks that previously required human input. Some examples of such applications include interactive voice response (IVR) systems and automated cheque-processing systems.

Natural language input is inherently inconsistent, noisy, and ambiguous, leading to potential recognition and decoding errors. However, high recognition accuracy is required for these applications to operate successfully, since mistakes can be expensive and frustrating to users. As a result, recognition systems must make use of as much contextual information as possible to increase the possibility of correctly recognizing the input. As an example, when recognizing input that must represent a country name, the recognition system can use a pre-defined list of valid country names to guide the recognition procedure. Similarly, when recognizing a phone number, a limited character set (i.e. digits) can be used to constrain the recognition results.

Contextual information used by recognition systems is generally programmed into the system by application developers. However, in some situations this is not possible due to the fact that the context is not known at the time of development, or the system is a legacy application that was not originally designed for use with a recognition system. This document discusses techniques that allow contextual information useful for speech and handwriting recognition to be automatically generated from documents, forms, and application data. In addition to this, techniques for tagging form fields with custom attributes and program code to aid recognition are discussed.

1.1 Digital Ink

Digital ink is a digital representation of the information generated by a pen-based input device. Generally, digital ink is structured as a sequence of strokes that begin when the pen device makes contact with a drawing surface and ends when the pen device is lifted. Each stroke comprises a set of sampled coordinates that define the movement of the pen whilst the pen is in contact with the drawing surface.

1.2 Handwriting Recognition

The goal of a handwriting recognition system is convert a sequence of written letters into computer text with the greatest possible accuracy. Handwriting recognition systems can be classed as:

- online, meaning that they work with dynamic writing information such as is generated by a pen computing system, or
- offline, meaning that that work with static representations of the writing such as a bitmap image.

The basic processing steps of an online handwriting recognition system are depicted in (similar processing occurs for offline systems). Processing begins when a pen device

generates a stream of digital ink that is to be recognized by the system. Usually, one or more pre-processing procedures are applied to this ink to remove sampling noise (e.g. jitter, spikes, etc.) and normalize the ink (e.g. removing slant, re-sampling to points, applying size normalization, etc.). The normalized ink is then segmented to produce a stream of basic elements required for the classification procedure (e.g. words, characters, or strokes). Note that often this segmentation is "soft", mean that a number of potential segmentation points are located, and the final segmentation points are resolved during classification or context processing.

The segmented ink is then passed to the classifier where it is used to generate letter hypotheses based on a pre-defined handwriting model. These hypotheses give an indication of the probability that a sequence of segments within the ink represent a basic language element (i.e. character or word). After classification, the context-processing module uses the language element hypotheses generated by the classifier to decode the ink according to a specified language model such as a dictionary or character grammar. The result produced by the context processing is passed to the application as the recognized text.

1.3 Form Processing

The basic processing required for online handwritten form recognition using form information for context processing is given in Figure 3. In this example, digital ink used to fill in the fields in a form is captured using an ink-recording device (e.g. digitising tablet, tablet computer, etc.) Each stroke in the digital ink is assigned to the appropriate field on the form using the form layout as defined in the form definition. The ink associated with each field is subsequently submitted to a handwriting recognition system that recognizes the ink. The recognition procedure uses context information that was derived from the field labels and attributes as specified in the form definition. When recognition is complete, the form data is passed to an application for processing.

Similarly, the processing required for offline handwritten form recognition using form information for context processing is given in Figure 4. In this example, a printed form is filled in using a normal writing instrument (e.g. pen or pencil) and is subsequently scanned into the system using a scanner or fax machine. The layout of the scanned form is analysed, and any machine-printed field labels found are converted to text using an Optical Character Recognition (OCR) system. The data entry fields are then recognized by an offline handwriting recognition system that uses the field labels located earlier to determine what context processing should be used.

1.4 Prior Work

Levitan [2] describes a form processing system that uses an optical recognition engine that "can recognize machine printed but not handwritten characters to locate the fields in the digital image by locating the machine printed field identifiers". When a field has been found, an "offline handwritten character recognition algorithm is then utilized to recognize the handwritten characters in each individual field". Finally, the field identifier and associated recognized handwritten text is stored in a database. Note that this patent does not discuss using the recognized field label to alter the context processing of the handwriting recognition; rather, the recognized text label is simply used opaquely as a key for data storage.

Bradley [3] uses a field-based recognition system to determine the best type of classifier (i.e. constrained handprint, unconstrained handprint, unconstrained cursive writing) to be used for each field on a form. The system uses confidence values and an adaptive weighting system to determine which type of classifier will generate the best results.

Anderson et al [4] describe "an advanced data capture architecture" that combines form definition capabilities with a character recognition processor. "It enables the user to freely define and redefine the format of document forms without requiring any reprogramming of the data processors which capture and use the data on the completed forms."

In [5], a system for the automatic analysis of web forms for the purpose of automated form fill-in is described. The software (called RoboForm) uses a Forms Description Language (FDL) to "describe a class of forms that a particular application of RoboForm can fill out." The system uses Artificial Intelligence (AI) techniques to deduce the field types that appear on a form.

2 Form Definition Information

This section describes techniques that can be used to improve recognition accuracy based on information contained within the definition and layout of a form. The use of field labels to determine field types is discussed, and a number of examples of context processing techniques based on field label identification are given. In addition to this, a discussion on using field types and attributes for improving recognition results is given, along with the use of custom fields and validation code.

Note that while handwriting recognition is used in many of the examples, the technique is suitable for use where ever potential ambiguity exists in the input. Thus, many of the techniques would also be applicable for speech recognition processing, etc.

2.1 Field Labels

Documents and forms that require data to be entered, either using a keyboard for screen-based applications or handwritten for pen computing or paper documents, must give the user some indication of the type of information that is required. This is usually done by labelling each data input area (or field) with a descriptive identifier, for example, "First Name", "Last Name", "Address", "Phone Number", and so on. For printed forms, this information appears as printed text on the paper, while online (i.e. computer-based) documents usually contain this information as a visible text entry defined in the structured description of the form.

The information contained in the field labels described above can be used to guide a handwriting recognition system. This is done by first associating each field label with the appropriate data entry region, either by analysing the form description (for online forms with a known format such as HTML [8], XHTML [10], PDF [7], or XML/XForms [6]) or by using document layout analysis [1] to associate labels with text entry regions, and OCR to interpret the label text (for forms of unknown format such as scanned forms or faxes). Once each label is associated with an entry field, a table of previously defined field label strings is searched (possibly using regular expression matching) and the corresponding recognition context information is found. This context information can then be used to improve the accuracy of handwriting recognition input.

Note that multiple field labels may map to each field type (e.g. "First Name", "Given Name", and "Firstname" will all map to the same context type). The following table gives example field label strings and the associated contextual information that can be used to improve the recognition results:

Field Label String	Context Processing
First Name, Given Name, etc.	Large lists of common first names are available for use as dictionaries during recognition. These lists, which are often derived from census data, include associated *a-priori* probabilities, allowing common names such as "John" and "David" to be more frequently matched. If additional information is available that indicates the gender of the writer, separate male and female lists can be used to further improve recognition accuracy. Note that during recognition, out-of-vocabulary words (i.e. names that do not appear in the name dictionary) must be allowed to ensure that unusual and uniquely spelled names can still be recognized correctly. This can be done by combining the dictionary decoding with a probabilistic grammar (such as an character n-gram) that contains information regarding the *a-priori* probability of character sequences usually found in names.

Last Name, Surname, Family Name, etc.	Similar to above, but using a dictionary of last names. Note that for Western names, there is generally much greater variability of last names across the population, so the probability of out-of-vocabulary words must be higher than that for first name recognition.
Address	Most addresses follow a regular pattern (e.g. dwelling number, followed by street name and street type). The recognition system can exploit this pattern during decoding by, for example, using regular expression matching, or by altering the valid character set (i.e. digits only, letters only, '/' allowed or not allowed, etc.) as recognition proceeds. In addition to this, some elements in the address can be decoded with the assistance of a dictionary, such as street type ("Street", "Road", "Place", "Avenue", etc.) or street names (common street names include "Main", "Church", "North", etc.)
Suburb, Town, etc.	Full lists of suburbs and towns are freely available for most regions. This information can be used in conjunction with state or postcode / zipcode information (if available) to further reduce the recognition alternatives.
State	Lists of states are available if the Country/Region is known. Each state can be given an *a-priori* probability corresponding in the likelihood that a person is from that state (i.e. large, populous states can be given a higher *a-priori* probability). Further constrains can be used if postcode / zipcode is known.
Phone Number	Phone numbers follow a regular pattern (e.g. "(##) ####-####") that can be used during recognition. Additionally, the valid character set for a phone number is constrained, further restricting the potential recognition alternatives.
Zip/Postal Code	Zip/Postal codes generally follow a specific pattern. Additional decoding constraints are available if the corresponding State and Suburb information is available.
Country, Region, etc.	Full lists of possible Country/Region labels are available, and are currently used by many web forms.
Birth Date, Date of Birth, etc.	Written dates generally follow a regular pattern, and have a constrained character set.
Email, E-Mail, Email Address, etc.	Email addresses follow a specific pattern and have a well-specified character set. An example regular expression that can be used to match email addresses is "/^([a-zA-Z0-9_\.\-])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9])+$/". In addition to this, if email contact information is available for a user (e.g. using Microsoft Windows Messaging API (MAPI)), the list of email addresses can be used as a dictionary during recognition. Similarly, common email domain names (e.g. "hotmail.com", "yahoo.com", "email.com", etc.) can be used as dictionary entries to guide recognition.
Credit Card, Credit Card Number, etc.	Credit card numbers have a specific format (e.g. "####-####-####-####") and constrained character set. Additionally, there are often validation rules (e.g. check digit tests) that can also be used during recognition. For example, if there are two equi-probable results for the recognition of a credit-card number, check digit validation may be of use in selecting the correct result.

Language / Locale	Lists of languages that are spoken around the world are freely available, and are currently used by many web forms. Once the language of a particular writer is known, it can be used to improve the processing of other types of input. Examples of this include different language-specific dictionaries (e.g. English, German, French, etc.) for text recognition, changing the valid recognition character set (e.g. allowing accented letters that are used by some Western European languages), and changing the format for date recognition.

In addition to using field-specific context processing, the field label can be used to store text that has been previously recognized for the specific field type. This text can be used as a dictionary entry in subsequent recognition of the same field type, since the user is likely to write the same value again (e.g. a user will usually write the same text each time a "First Name" field must be completed).

Similarly, information that the processing system has stored in a database for a user (e.g. the system may know their name, address, date-of-birth, etc.) can also be used as dictionary entries for form processing by associating the database field label with the form field label (e.g. the "First Name" text in the database can be used as a dictionary entry for a "First Name" form field that requires recognition).

2.2 Field Attributes

Most form definition formats support a number of different field types, such as text fields, selection list fields, combination fields (i.e. a field that combines text input with a selection list), signature fields, checkboxes, buttons, and so on. The field type gives some indication of the expected input data-type (e.g. a text input field indicates text entry). If a document format allows data-types to be explicitly defined (e.g. XML/XForms), a recognition system can use this information to constrain the recognition process.

In addition to the field type, forms often contain information regarding the type of data that should be entered in each field. This information is usually contained in attributes that are associated with a specific field. One example of this is the set of selection strings that are commonly associated with list input fields. These strings represent the options from which the user must make a selection, and can be used as dictionary elements during recognition. Similarly, recognition of combination fields can use a dictionary of selection strings in combination with a character grammar to allow words other than those listed in the option list to be recognized.

Standard input fields may also contain attributes that can assist in the recognition procedure. For example, some input field types have a flag indicating that the value entered must be numeric, signifying to the recognition system that the recognized character set should only include digits. Input fields may also contain a mask attribute, which is a string indicating that the input must match the specified pattern (e.g. "####AA" requiring that four digits followed by two upper-case alphabetic letters be entered such as "2002CY"). This mask can be used to constrain the valid recognition character set at each offset in the string and thus improve the recognition accuracy.

Many forms specify validation parameters that can be used to guide the recognition process. For example, numeric input fields may specify minimum and maximum values that can be used to constrain the recognition results. Other fields may contain validation program code (e.g. JavaScript [9]) that is executed when the user has entered a value into the field. This code can be executed multiple times, with each individual recognition result as a parameter, allowing potential alternative results that do not conform to the validation requirements to be discarded.

2.3 Custom Attributes

In addition to using standard form field attributes to improve the recognition process, recognition-specific information can be added to fields using custom attributes. This information is only used if the form input is processed using a recognition system. Thus, the

form can still be used normally where required (e.g. data entry using a keyboard via a web browser) since the custom attributes are ignored; however, if recognition is required, the custom parameters can be used to improve the recognition results.

Some examples of custom field attributes include character set definition (where the set of valid characters for a field is explicitly defined) and regular expressions. If the fields are displayed or printed using visual cues to guide character spacing (e.g. boxes on forms where each box must contain a single character), the parameters of the guide can be associated with the field as custom attributes to assist with the character segmentation stage of the handwriting recognition. For example, by specifying the coordinates of the bounding rectangle and the number of rows and columns in a field that uses character boxes for input, the recognition system can be informed of the expected location of each character, allowing more accurate recognition to occur.

Information regarding context processing and language modelling can also be encoded in custom attributes. Some handwriting recognition systems use a combination of language models to assist in the recognition of handwritten text (e.g. n-gram character models, standard dictionaries, user-specific dictionaries, etc.). These models are usually combined using a set of weightings that indicate the likelihood that an input word will be decoded correctly using each of the specified models. However, the most accurate results are produced when the weightings can be customized depending on the expected input. By including the language model weights as a custom attribute for a field, more accurate recognition can be achieved by tuning the model weights on a per form or even per field basis.

2.4 Custom Validation Code

To allow more control over the recognition procedure, custom validation program code (e.g. JavaScript) can be associated with a field that is executed on each potential result after the handwriting recognition procedure has completed, allowing the most appropriate result to be selected. However, rather than using a Boolean validation function (i.e. a string is either valid or invalid), the function can return a confidence value that indicates the probability that the string would be entered. This probability can be combined with the results of the character classification procedure to select the most probable recognition result.

An improvement to this scheme is to define a language model probability function that is called by the recogniser as each character is recognized by the system. This allows a recognition system to prune unlikely or invalid recognition string early in the recognition procedure, allowing long strings of text to be recognized efficiently. During the recognition procedure, a large number of potential results are produced by considering different combinations of recognized characters. Typically, there are a large number of potential character alternatives for each letter position, so for text of even moderate length, there are a large number of alternatives. As a result, recognition systems generally use a beam search technique, such that the n best alternatives at each letter position are considered, where n is typically between 10 and 100. Thus, the n most likely results at each position are stored, with the remainder discarded.

However, to select the n best results at each step requires validation from the language model at each step rather than after the recognition procedure has completed, otherwise high-scoring strings that are impossible or unlikely as defined by the language model may be retained while valid but lower-scoring strings are discarded. As a result, the improved language model function should be able to calculate and return a sub-string probability, so that the recogniser can combine the character classification probability with the sub-string probability at each step, and thus select the n most likely strings. This flexible approach allows almost any language model, including dictionaries and character Markov-models, to be implemented.

3 Document Format Examples

This section describes the extraction and use of form definition information from a number of commonly used document formats, including HTML, XForms, and PDF. This information can be subsequently used to improve the accuracy of a handwriting recognition system. Also discussed are ways to add custom attributes to the document formats, allowing recognition-specific information to be included in the form definition.

3.1 Hypertext Mark-up Language (HTML/XHTML)

Hypertext Mark-up Language (HTML) is a standard set of mark-up symbols used to define the format of a page of text and graphics that is intended for display in a World Wide Web browser. HTML is a formal recommendation by the World Wide Web Consortium (W3C) and is defined in [8]. XHTML, a reformulation of HTML as an XML application, is very similar to HTML and is defined in [10] , and similarly, SGML [11]. Some example HTML code for a form is given below (an example of the output that this code might generate in a browser is given in Figure 1):

```
<html>
<form ACTION="cgi-bin/form.exe" METHOD=post>
<p><b>Please Enter Your Name</b></p>
<p>First Name: <INPUT TYPE="TEXT" NAME="FirstName" CUSTOM="Hello"></p>
<p>Last Name:  <INPUT TYPE="TEXT" NAME="LastName"></p>
<p><INPUT TYPE="SUBMIT" NAME="Submit"></p>
</form>
</html>
```

Usually, field labels associated with input fields can be easily derived from the HTML document source. Generally, field labels appear as normal text immediately before the input field definition (as shown above). In other situations, the layout of the rendered document can be analysed to determine which text labels should be associated with which input fields (for example, when a table is used for form layout). Additionally, the "name" attribute that is associated with many input elements may contain text that will allow the field type to be determined.

Standard HTML contains a number of element attributes that can be usefully used as hints to a recognition system. Some examples include:

- the "maxlength" attribute of an INPUT element that can be used to limit the length of the recognized text,
- the OPTION elements associated with a SELECT element that represent the set of valid input strings (which can be used as dictionary entries during recognition), and
- the "rows" and "cols" attributes in a TEXTAREA element that could be used to define a character spacing guide (e.g. boxed input where each letter must be written in a separate box).

In addition to this, custom attributes can be easily added to HTML field elements (e.g. CUSTOM="Hello"), since browsers and other systems processing a page must ignore attributes that are unknown.

3.2 XML/XForms

XFORMS is a standard form definition language defined by W3C and described in [6]. The XForms standard has been developed as a successor to HTML forms, and implements device independent form processing by allowing the same form to operate on desktop computers, hand-held devices, information appliances, and even paper. To achieve this, XForms ensures that, unlike HTML, data definitions are kept sperate from presentation. An example of XForms code is given below (an example of the output that this code might generate in a browser is given in Figure 2):

```
<xform>
<submitInfo action='form.exe' method='post'/>
</xform>

<input xform='payment' ref='cc'>
  <caption>Credit Card Number</caption>
</input><input xform='payment' ref='exp'>
  <caption>Expiration Date</caption>
</input><submit xform='payment'>
  <caption>Submit</caption>
</submit>
```

In a similar manner to HTML, field labels can be derived from the XForms code by examining text elements immediately prior to the input field definitions. However, this process is easier in XForms since field labels are preceded by a "<caption>" tag allowing labels to be unambiguously identified. In addition to this, XForms supports input field elements similar to those described previously for HTML, including the list selection elements "<selectOne>" and "<selectMany>" and associated "<item>" elements that can be used a dictionary entries during recognition processing.

The XForms specification includes a set of data-types for field input, including date, money, number, string, time, and URI types. This information can be used by a recognition system to improve recognition accuracy. Similarly, the specification includes data attributes (e.g. currency, decimal places, integer) and validation attributes (minimum value, maximum value, pattern, range), which can be used to further improve recognition results.

3.3 Portable Document Format (PDF)

Portable Document Format (PDF) is a document format defined by Adobe [7] that has become the de-facto standard for internet-based document distribution. Recently, Adobe has added interactive elements that allow the definition of forms for online use.

Like HTML and XForms, PDF form elements have a specific type (e.g. text, signature, combo box, list box) that defines the behaviour of the element and thus can be used as a guide for a handwriting recognition system. They also contain a field name (e.g. "/T (FirstName)") that may contain a useful label that indicates the type of data to be entered into the field. List and combination fields contain a set of options ("/Opt [(Option1)(Option2)]") that define the valid selection strings.

Additional field attributes include a format specifier (e.g. number, percent, date, time, zip code, phone number, social security number, etc.) and JavaScript validation code that is executed when data has been entered into the field. Custom attributes can also be easily incorporated in field definitions, as shown above ("/CUSTOM_ATTRIBUTE (HelloWorld)").

4 Conclusion

This document has discussed techniques for deriving information from the definition and layout of a form for use with a handwriting recognition system. The novel aspects of this discussion include:

- using the descriptive identifier (field label) associated with an input field to determine the type of context processing that should be used when recognizing the field,
- deriving fields label from the document definition (for online documents) or with document layout analysis and OCR (from scanned forms or faxes),
- using field attributes and validation parameters specified in a document definition to assist the recognition procedure,
- associating recognition-specific information with fields using custom attributes,
- field attributes that contain program code that is used during the handwriting recognition process to model language constraints, and
- using the above techniques to process HTML, XHTML, SGML, PDF, and XML/XForms documents for use with a recognition system.

5 Figures

Figure 1. Example HTML Form Output

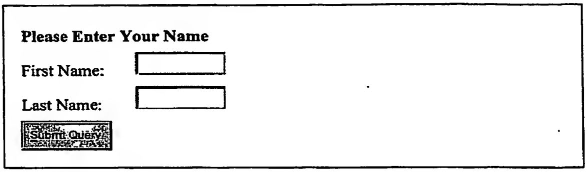


Figure 2. XML/XForms Form Example

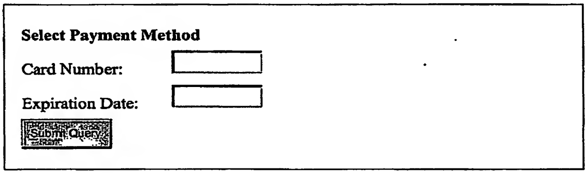


Figure 3. Online Form Processing

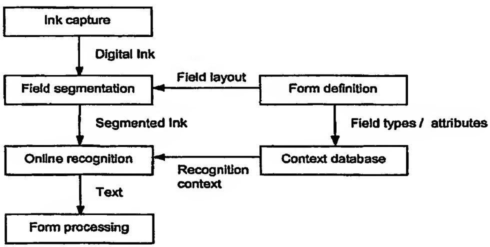


Figure 4. Offline Form Processing

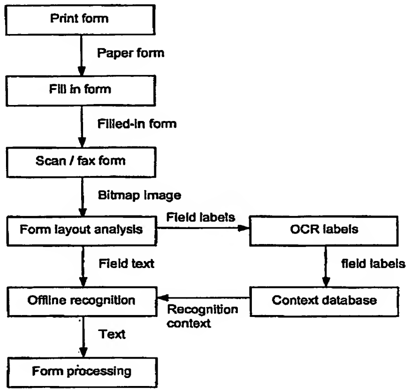
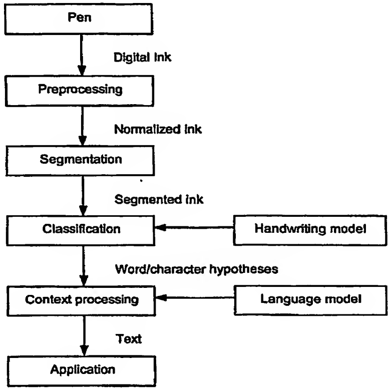


Figure 5. Handwriting Recognition Process



6 References

[1] T. Watanabe, H. Naruse, Q. Luo, and N. Sugie, "Structure Analysis of Table-Form Documents on the Basis of the Recognition of Vertical and Horizontal Line Segments", ICDAR-91, pp. 638-646, 1991.

[2] Levitan. A, "System and method for automatic optical data entry", US Patent 5,237,628, August 17, 1993.

[3] Bradley, M., "System and method for enhanced character recognition accuracy by adaptive probability weighting", US Patent 5,455,872, October 3, 1995.

[4] Anderson, G. et. al., "Advanced data capture architecture data processing system and method for scanned images of document forms", US Patent 5,235,654, August 10, 1993.

[5] Siber Systems, "RoboForm Feature Description", http://www.roboform.com/ai.html, August 1, 2002.

[6] World Wide Web Consortium, "XForms 1.0", W3C Working Draft 21, August 2002.

[7] Adobe Systems Incorporated, "Acrobat Core API Reference", Version Acrobat 5.0, December 2001.

[8] World Wide Web Consortium (W3C), "HTML 4.01 Specification", W3C Recommendation, 24 December 1999.

[9] Netscape Communications Corporation, "Core JavaScript Reference", 1998.

[10] World Wide Web Consortium (W3C), "XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)", W3C Recommendation, 1 August 2002.

[11] International Organization for Standardization (ISO), "Information processing - Text and office systems - Standard Generalized Markup Language (SGML)", ISO 8879, 1986.